autogluon_each_location

October 16, 2023

[1]:
```
# config

label = 'y'
metric = 'mean_absolute_error'
time_limit = 60*30
presets = 'best_quality'

do_drop_ds = True
# hour, dayofweek, dayofmonth, month, year
use_dt_attrs = []#["hour", "year"]
use_estimated_diff_attr = False
use_is_estimated_attr = True

use_groups = False
n_groups = 8

auto_stack = True
num_stack_levels = 1
num_bag_folds = 8


use_tune_data = False
use_test_data = True
tune_and_test_length = 24*30*3 # 3 months from end
holdout_frac = None
use_bag_holdout = False # Enable this if there is a large gap between score_val␣
↪and score_test in stack models.

sample_weight = 'sample_weight' #None
weight_evaluation = True
sample_weight_estimated = 3

run_analysis = True
```

[2]:
```
import pandas as pd
import numpy as np
```

```python
import warnings
warnings.filterwarnings("ignore")


def feature_engineering(X):
    # shift all columns with "1h" in them by 1 hour, so that for index 16:00,␣
↪we have the values from 17:00
    # but only for the columns with "1h" in the name
    #X_shifted = X.filter(regex="\dh").shift(-1, axis=1)
    #print(f"Number of columns with 1h in name: {X_shifted.columns}")


    columns = ['clear_sky_energy_1h:J', 'diffuse_rad_1h:J', 'direct_rad_1h:J',
       'fresh_snow_12h:cm', 'fresh_snow_1h:cm', 'fresh_snow_24h:cm',
       'fresh_snow_3h:cm', 'fresh_snow_6h:cm']

    X_shifted = X[X.index.minute==0][columns].copy()
    # loop through all rows and check if index + 1 hour is in the index, if so␣
↪get that value, else nan
    count1 = 0
    count2 = 0
    for i in range(len(X_shifted)):
        if X_shifted.index[i] + pd.Timedelta('1 hour') in X.index:
            count1 += 1
            X_shifted.iloc[i] = X.loc[X_shifted.index[i] + pd.Timedelta('1␣
↪hour')][columns]
        else:
            count2 += 1
            X_shifted.iloc[i] = np.nan

    print("COUNT1", count1)
    print("COUNT2", count2)

    X_old_unshifted = X[X.index.minute==0][columns]
    # rename X_old_unshifted columns to have _not_shifted at the end
    X_old_unshifted.columns = [f"{col}_not_shifted" for col in X_old_unshifted.
↪columns]

    # put the shifted columns back into the original dataframe
    #X[columns] = X_shifted[columns]



    date_calc = None
    if "date_calc" in X.columns:
```

```
        date_calc = X[X.index.minute == 0]['date_calc']

    # resample to hourly
    X = X.resample('H').mean()

    X[columns] = X_shifted[columns]
    #X[X_old_unshifted.columns] = X_old_unshifted

    if date_calc is not None:
        X['date_calc'] = date_calc

    return X




def fix_X(X, name):
    # Convert 'date_forecast' to datetime format and replace original column␣
 ↪with 'ds'
    X['ds'] = pd.to_datetime(X['date_forecast'])
    X.drop(columns=['date_forecast'], inplace=True, errors='ignore')
    X.sort_values(by='ds', inplace=True)
    X.set_index('ds', inplace=True)


    X = feature_engineering(X)

    return X



def handle_features(X_train_observed, X_train_estimated, X_test, y_train):
    X_train_observed = fix_X(X_train_observed, "X_train_observed")
    X_train_estimated = fix_X(X_train_estimated, "X_train_estimated")
    X_test = fix_X(X_test, "X_test")


    # add sample weights, which are 1 for observed and 3 for estimated
    X_train_observed["sample_weight"] = 1
    X_train_estimated["sample_weight"] = sample_weight_estimated
    X_test["sample_weight"] = sample_weight_estimated


    y_train['ds'] = pd.to_datetime(y_train['time'])
    y_train.drop(columns=['time'], inplace=True)
    y_train.sort_values(by='ds', inplace=True)
    y_train.set_index('ds', inplace=True)
```

```
    return X_train_observed, X_train_estimated, X_test, y_train




def preprocess_data(X_train_observed, X_train_estimated, X_test, y_train,␣
↪location):
    # convert to datetime
    X_train_observed, X_train_estimated, X_test, y_train =␣
↪handle_features(X_train_observed, X_train_estimated, X_test, y_train)

    if use_estimated_diff_attr:
        X_train_observed["estimated_diff_hours"] = 0
        X_train_estimated["estimated_diff_hours"] = (X_train_estimated.index -␣
↪pd.to_datetime(X_train_estimated["date_calc"])).dt.total_seconds() / 3600
        X_test["estimated_diff_hours"] = (X_test.index - pd.
↪to_datetime(X_test["date_calc"])).dt.total_seconds() / 3600

        X_train_estimated["estimated_diff_hours"] =␣
↪X_train_estimated["estimated_diff_hours"].astype('int64')
        # the filled once will get dropped later anyways, when we drop y nans
        X_test["estimated_diff_hours"] = X_test["estimated_diff_hours"].
↪fillna(-50).astype('int64')

    if use_is_estimated_attr:
        X_train_observed["is_estimated"] = 0
        X_train_estimated["is_estimated"] = 1
        X_test["is_estimated"] = 1

    # drop date_calc
    X_train_estimated.drop(columns=['date_calc'], inplace=True)
    X_test.drop(columns=['date_calc'], inplace=True)


    y_train["y"] = y_train["pv_measurement"].astype('float64')
    y_train.drop(columns=['pv_measurement'], inplace=True)
    X_train = pd.concat([X_train_observed, X_train_estimated])


    # clip all y values to 0 if negative
    y_train["y"] = y_train["y"].clip(lower=0)

    X_train = pd.merge(X_train, y_train, how="inner", left_index=True,␣
↪right_index=True)
```

```python
    # print number of nans in sample_weight
    print(f"Number of nans in sample_weight: {X_train['sample_weight'].isna().
 ↪sum()}")
    # print number of nans in y
    print(f"Number of nans in y: {X_train['y'].isna().sum()}")


    X_train["location"] = location
    X_test["location"] = location

    return X_train, X_test
# Define locations
locations = ['A', 'B', 'C']

X_trains = []
X_tests = []
# Loop through locations
for loc in locations:
    print(f"Processing location {loc}...")
    # Read target training data
    y_train = pd.read_parquet(f'{loc}/train_targets.parquet')

    # Read estimated training data and add location feature
    X_train_estimated = pd.read_parquet(f'{loc}/X_train_estimated.parquet')

    # Read observed training data and add location feature
    X_train_observed= pd.read_parquet(f'{loc}/X_train_observed.parquet')

    # Read estimated test data and add location feature
    X_test_estimated = pd.read_parquet(f'{loc}/X_test_estimated.parquet')

    # Preprocess data
    X_train, X_test = preprocess_data(X_train_observed, X_train_estimated,␣
 ↪X_test_estimated, y_train, loc)

    X_trains.append(X_train)
    X_tests.append(X_test)

# Concatenate all data and save to csv
X_train = pd.concat(X_trains)
X_test = pd.concat(X_tests)
```

```
Processing location A…
COUNT1 29667
COUNT2 1
COUNT1 4392
COUNT2 2
```

```
COUNT1 702
COUNT2 18
Number of nans in sample_weight: 0
Number of nans in y: 0
Processing location B…
COUNT1 29232
COUNT2 1
COUNT1 4392
COUNT2 2
COUNT1 702
COUNT2 18
Number of nans in sample_weight: 0
Number of nans in y: 4
Processing location C…
COUNT1 29206
COUNT2 1
COUNT1 4392
COUNT2 2
COUNT1 702
COUNT2 18
Number of nans in sample_weight: 0
Number of nans in y: 6059
```

1 Feature enginering

[3]:
```
import numpy as np
import pandas as pd

X_train.dropna(subset=['y'], inplace=True)



for attr in use_dt_attrs:
    X_train[attr] = getattr(X_train.index, attr)
    X_test[attr] = getattr(X_test.index, attr)

print(X_train.head())





if use_groups:
    # fix groups for cross validation
    locations = X_train['location'].unique()  # Assuming 'location' is the name␣
 ↪of the column representing locations

    grouped_dfs = []  # To store data frames split by location
```

```
    # Loop through each unique location
    for loc in locations:
        loc_df = X_train[X_train['location'] == loc]

        # Sort the DataFrame for this location by the time column
        loc_df = loc_df.sort_index()

        # Calculate the size of each group for this location
        group_size = len(loc_df) // n_groups

        # Create a new 'group' column for this location
        loc_df['group'] = np.repeat(range(n_groups),␣
 ↪repeats=[group_size]*(n_groups-1) + [len(loc_df) - group_size*(n_groups-1)])

        # Append to list of grouped DataFrames
        grouped_dfs.append(loc_df)

    # Concatenate all the grouped DataFrames back together
    X_train = pd.concat(grouped_dfs)
    X_train.sort_index(inplace=True)
    print(X_train["group"].head())




to_drop = ["snow_drift:idx", "snow_density:kgm3", "wind_speed_w_1000hPa:ms",␣
 ↪"dew_or_rime:idx", "prob_rime:p", "fresh_snow_12h:cm", "fresh_snow_24h:cm"]

X_train.drop(columns=to_drop, inplace=True)
X_test.drop(columns=to_drop, inplace=True)

X_train.to_csv('X_train_raw.csv', index=True)
X_test.to_csv('X_test_raw.csv', index=True)
```

```
                     absolute_humidity_2m:gm3  air_density_2m:kgm3  \
ds
2019-06-02 22:00:00                     7.700              1.22825
2019-06-02 23:00:00                     7.700              1.22350
2019-06-03 00:00:00                     7.875              1.21975
2019-06-03 01:00:00                     8.425              1.21800
2019-06-03 02:00:00                     8.950              1.21800

                     ceiling_height_agl:m  clear_sky_energy_1h:J  \
ds
2019-06-02 22:00:00           1728.949951               0.000000
2019-06-02 23:00:00           1689.824951               0.000000
```

```
2019-06-03 00:00:00          1563.224976           0.000000
2019-06-03 01:00:00          1283.425049        6546.899902
2019-06-03 02:00:00          1003.500000      102225.898438

                     clear_sky_rad:W  cloud_base_agl:m  dew_or_rime:idx  \
ds
2019-06-02 22:00:00             0.00       1728.949951              0.0
2019-06-02 23:00:00             0.00       1689.824951              0.0
2019-06-03 00:00:00             0.00       1563.224976              0.0
2019-06-03 01:00:00             0.75       1283.425049              0.0
2019-06-03 02:00:00            23.10       1003.500000              0.0

                     dew_point_2m:K  diffuse_rad:W  diffuse_rad_1h:J  …  \
ds                                                                    …
2019-06-02 22:00:00      280.299988          0.000          0.000000  …
2019-06-02 23:00:00      280.299988          0.000          0.000000  …
2019-06-03 00:00:00      280.649994          0.000          0.000000  …
2019-06-03 01:00:00      281.674988          0.300       7743.299805  …
2019-06-03 02:00:00      282.500000         11.975      60137.601562  …

                     total_cloud_cover:p  visibility:m  wind_speed_10m:ms  \
ds
2019-06-02 22:00:00           100.000000  40386.476562              3.600
2019-06-02 23:00:00           100.000000  33770.648438              3.350
2019-06-03 00:00:00           100.000000  13595.500000              3.050
2019-06-03 01:00:00           100.000000   2321.850098              2.725
2019-06-03 02:00:00            99.224998  11634.799805              2.550

                     wind_speed_u_10m:ms  wind_speed_v_10m:ms  \
ds
2019-06-02 22:00:00               -3.575               -0.500
2019-06-02 23:00:00               -3.350                0.275
2019-06-03 00:00:00               -2.950                0.750
2019-06-03 01:00:00               -2.600                0.875
2019-06-03 02:00:00               -2.350                0.925

                     wind_speed_w_1000hPa:ms  sample_weight  is_estimated  \
ds
2019-06-02 22:00:00                      0.0              1             0
2019-06-02 23:00:00                      0.0              1             0
2019-06-03 00:00:00                      0.0              1             0
2019-06-03 01:00:00                      0.0              1             0
2019-06-03 02:00:00                      0.0              1             0

                         y  location
ds
2019-06-02 22:00:00   0.00         A
2019-06-02 23:00:00   0.00         A
```

```
2019-06-03 00:00:00   0.00         A
2019-06-03 01:00:00   0.00         A
2019-06-03 02:00:00  19.36         A

[5 rows x 49 columns]
```

[4]:
```
from autogluon.tabular import TabularDataset, TabularPredictor
from autogluon.timeseries import TimeSeriesDataFrame
import numpy as np
train_data = TabularDataset('X_train_raw.csv')
# set group column of train_data be increasing from 0 to 7 based on time, the␣
↪first 1/8 of the data is group 0, the second 1/8 of the data is group 1, etc.
train_data['ds'] = pd.to_datetime(train_data['ds'])
train_data = train_data.sort_values(by='ds')

# # print size of the group for each location
# for loc in locations:
#     print(f"Location {loc}:")
#     print(train_data[train_data["location"] == loc].groupby('group').size())


# get end date of train data and subtract 3 months
split_time = pd.to_datetime(train_data["ds"]).max() - pd.
↪Timedelta(hours=tune_and_test_length)
train_set = TabularDataset(train_data[train_data["ds"] < split_time])
test_set = TabularDataset(train_data[train_data["ds"] >= split_time])
if use_groups:
    test_set = test_set.drop(columns=['group'])

if do_drop_ds:
    train_set = train_set.drop(columns=['ds'])
    test_set = test_set.drop(columns=['ds'])
    train_data = train_data.drop(columns=['ds'])


def normalize_sample_weights_per_location(df):
    for loc in locations:
        loc_df = df[df["location"] == loc]
        loc_df["sample_weight"] = loc_df["sample_weight"] /␣
↪loc_df["sample_weight"].sum() * loc_df.shape[0]
        df[df["location"] == loc] = loc_df
    return df

tuning_data = None
if use_tune_data:
    train_data = train_set
    if use_test_data:
```

```
        # split test_set in half, use first half for tuning
        tuning_data, test_data = [], []
        for loc in locations:
            loc_test_set = test_set[test_set["location"] == loc]
            loc_tuning_data = loc_test_set.iloc[:len(loc_test_set)//2]
            loc_test_data = loc_test_set.iloc[len(loc_test_set)//2:]
            tuning_data.append(loc_tuning_data)
            test_data.append(loc_test_data)
        tuning_data = pd.concat(tuning_data)
        test_data = pd.concat(test_data)
        print("Shapes of tuning and test", tuning_data.shape[0], test_data.
 ↪shape[0], tuning_data.shape[0] + test_data.shape[0])

    else:
        tuning_data = test_set
        print("Shape of tuning", tuning_data.shape[0])

    # ensure sample weights for your tuning data sum to the number of rows in␣
 ↪the tuning data.
    tuning_data = normalize_sample_weights_per_location(tuning_data)


else:
    if use_test_data:
        train_data = train_set
        test_data = test_set
        print("Shape of test", test_data.shape[0])

# ensure sample weights for your training (or tuning) data sum to the number of␣
 ↪rows in the training (or tuning) data.
train_data = normalize_sample_weights_per_location(train_data)
if use_test_data:
    test_data = normalize_sample_weights_per_location(test_data)
```

```
Shape of test 5791
```

[5]:
```
if run_analysis:
    import autogluon.eda.auto as auto
    auto.dataset_overview(train_data=train_data, test_data=test_data,␣
 ↪label="y", sample=None)
```

train_data dataset summary

```
                              count  unique top   freq             mean  \
absolute_humidity_2m:gm3      87160     757                    6.138632
air_density_2m:kgm3           87160    1370                    1.253802
ceiling_height_agl:m          72139   59833                 2864.542561
clear_sky_energy_1h:J         87157   45557               518241.825283
```

clear_sky_rad:W	87160	19511			143.951884
cloud_base_agl:m	81279	61233			1736.160546
dew_point_2m:K	87160	2001			275.537267
diffuse_rad:W	87160	10980			39.210491
diffuse_rad_1h:J	87157	45515			141514.615214
direct_rad:W	87160	13914			50.139922
direct_rad_1h:J	87157	39280			180360.846534
effective_cloud_cover:p	87160	5652			67.118857
elevation:m	87160	3			11.411014
fresh_snow_1h:cm	87157	39			0.008136
fresh_snow_3h:cm	87157	68			0.024312
fresh_snow_6h:cm	87157	94			0.048424
is_day:idx	87160	5			0.482965
is_estimated	87232	2			0.05968
is_in_shadow:idx	87160	5			0.564895
location	87232	3	A	31924	
msl_pressure:hPa	87160	3693			1009.291473
precip_5min:mm	87160	270			0.005788
precip_type_5min:idx	87160	15			0.084236
pressure_100m:hPa	87160	3705			995.621975
pressure_50m:hPa	87160	3758			1001.745166
rain_water:kgm2	87160	39			0.010136
relative_humidity_1000hPa:p	87160	3787			73.860635
sample_weight	87232	6			1.0
sfc_pressure:hPa	87160	3780			1007.89561
snow_depth:cm	87160	483			0.197251
snow_melt_10min:mm	87160	63			0.000245
snow_water:kgm2	87160	161			0.09109
sun_azimuth:d	87160	82801			179.660078
sun_elevation:d	87160	71854			-1.225457
super_cooled_liquid_water:kgm2	87160	53			0.058341
t_1000hPa:K	87160	1986			279.712685
total_cloud_cover:p	87160	5546			73.819247
visibility:m	87160	85645			33233.674454
wind_speed_10m:ms	87160	594			3.025581
wind_speed_u_10m:ms	87160	988			0.664335
wind_speed_v_10m:ms	87160	848			0.694845
y	87232	10750			287.954185

	std	min	25% \
absolute_humidity_2m:gm3	2.73761	0.5	4.1
air_density_2m:kgm3	0.036657	1.13925	1.22875
ceiling_height_agl:m	2531.428872	27.8	1085.2625
clear_sky_energy_1h:J	828416.074463	0.0	0.0
clear_sky_rad:W	230.149085	0.0	0.0
cloud_base_agl:m	1797.954658	27.8	598.2875
dew_point_2m:K	6.846723	247.425	271.0
diffuse_rad:W	60.603659	0.0	0.0

```
diffuse_rad_1h:J                 216225.961408       0.0          0.0
direct_rad:W                         113.07899       0.0          0.0
direct_rad_1h:J                   402277.99935       0.0          0.0
effective_cloud_cover:p              34.037938       0.0     42.41875
elevation:m                           7.881548       6.0          6.0
fresh_snow_1h:cm                      0.107503       0.0          0.0
fresh_snow_3h:cm                      0.267725       0.0          0.0
fresh_snow_6h:cm                      0.457725       0.0          0.0
is_day:idx                            0.485944       0.0          0.0
is_estimated                          0.236894       0.0          0.0
is_in_shadow:idx                      0.483131       0.0          0.0
location
msl_pressure:hPa                     12.998509   944.375     1001.275
precip_5min:mm                        0.029771       0.0          0.0
precip_type_5min:idx                  0.325388       0.0          0.0
pressure_100m:hPa                    12.924683   929.975    987.69995
pressure_50m:hPa                     12.982562    935.75       993.75
rain_water:kgm2                       0.042332       0.0          0.0
relative_humidity_1000hPa:p          14.160229    19.575       64.425
sample_weight                         0.422269  0.876118     0.876118
sfc_pressure:hPa                     13.042592    941.55       999.85
snow_depth:cm                         1.284395       0.0          0.0
snow_melt_10min:mm                    0.003958       0.0          0.0
snow_water:kgm2                       0.240712       0.0          0.0
sun_azimuth:d                        97.308971     6.983     94.72475
sun_elevation:d                      24.168008   -49.932   -18.737563
super_cooled_liquid_water:kgm2        0.106882       0.0          0.0
t_1000hPa:K                           6.559438   258.025       275.15
total_cloud_cover:p                  33.768818       0.0       53.725
visibility:m                      18089.724083   132.375  16688.61875
wind_speed_10m:ms                     1.752114     0.025         1.65
wind_speed_u_10m:ms                   2.779236    -7.225        -1.35
wind_speed_v_10m:ms                   1.881059      -8.4        -0.55
y                                   766.111697       0.0          0.0

                                      50%         75%          max   dtypes  \
absolute_humidity_2m:gm3              5.6         8.0        17.35  float64
air_density_2m:kgm3                1.2525     1.27675        1.441  float64
ceiling_height_agl:m            1859.4751  3925.32485    12285.775  float64
clear_sky_energy_1h:J              4312.0    777195.2    3006697.2  float64
clear_sky_rad:W                       1.6       216.2       835.65  float64
cloud_base_agl:m                 1178.425   2081.0375    11673.725  float64
dew_point_2m:K                      275.4       280.8      293.625  float64
diffuse_rad:W                       0.875      64.325       334.75  float64
diffuse_rad_1h:J                   9534.0    233004.8    1182265.4  float64
direct_rad:W                          0.0      28.925        683.4  float64
direct_rad_1h:J                       0.0    111408.5    2445897.0  float64
effective_cloud_cover:p            79.675      98.475        100.0  float64
```

elevation:m	7.0	24.0	24.0	float64
fresh_snow_1h:cm	0.0	0.0	7.1	float64
fresh_snow_3h:cm	0.0	0.0	20.6	float64
fresh_snow_6h:cm	0.0	0.0	34.0	float64
is_day:idx	0.25	1.0	1.0	float64
is_estimated	0.0	0.0	1.0	int64
is_in_shadow:idx	1.0	1.0	1.0	float64
location				object
msl_pressure:hPa	1010.275	1018.35	1044.1	float64
precip_5min:mm	0.0	0.0	0.6225	float64
precip_type_5min:idx	0.0	0.0	5.0	float64
pressure_100m:hPa	996.7	1004.7	1030.875	float64
pressure_50m:hPa	1002.8	1010.825	1037.25	float64
rain_water:kgm2	0.0	0.0	1.1	float64
relative_humidity_1000hPa:p	76.2	85.25	100.0	float64
sample_weight	0.899142	0.907248	2.721744	float64
sfc_pressure:hPa	1008.925	1017.0	1043.725	float64
snow_depth:cm	0.0	0.0	18.2	float64
snow_melt_10min:mm	0.0	0.0	0.18	float64
snow_water:kgm2	0.0	0.1	5.65	float64
sun_azimuth:d	180.007	264.513	348.48752	float64
sun_elevation:d	-0.8645	15.234063	49.94375	float64
super_cooled_liquid_water:kgm2	0.0	0.1	1.375	float64
t_1000hPa:K	279.075	284.25	303.25	float64
total_cloud_cover:p	92.85	99.9	100.0	float64
visibility:m	37320.0515	48663.7615	75489.33	float64
wind_speed_10m:ms	2.7	4.05	13.275	float64
wind_speed_u_10m:ms	0.3	2.475	11.2	float64
wind_speed_v_10m:ms	0.725	1.875	8.825	float64
y	0.0	176.4	5733.42	float64

	missing_count	missing_ratio	raw_type	\
absolute_humidity_2m:gm3	72	0.000825	float	
air_density_2m:kgm3	72	0.000825	float	
ceiling_height_agl:m	15093	0.173021	float	
clear_sky_energy_1h:J	75	0.00086	float	
clear_sky_rad:W	72	0.000825	float	
cloud_base_agl:m	5953	0.068243	float	
dew_point_2m:K	72	0.000825	float	
diffuse_rad:W	72	0.000825	float	
diffuse_rad_1h:J	75	0.00086	float	
direct_rad:W	72	0.000825	float	
direct_rad_1h:J	75	0.00086	float	
effective_cloud_cover:p	72	0.000825	float	
elevation:m	72	0.000825	float	
fresh_snow_1h:cm	75	0.00086	float	
fresh_snow_3h:cm	75	0.00086	float	
fresh_snow_6h:cm	75	0.00086	float	

is_day:idx	72	0.000825	float
is_estimated			int
is_in_shadow:idx	72	0.000825	float
location			object
msl_pressure:hPa	72	0.000825	float
precip_5min:mm	72	0.000825	float
precip_type_5min:idx	72	0.000825	float
pressure_100m:hPa	72	0.000825	float
pressure_50m:hPa	72	0.000825	float
rain_water:kgm2	72	0.000825	float
relative_humidity_1000hPa:p	72	0.000825	float
sample_weight			float
sfc_pressure:hPa	72	0.000825	float
snow_depth:cm	72	0.000825	float
snow_melt_10min:mm	72	0.000825	float
snow_water:kgm2	72	0.000825	float
sun_azimuth:d	72	0.000825	float
sun_elevation:d	72	0.000825	float
super_cooled_liquid_water:kgm2	72	0.000825	float
t_1000hPa:K	72	0.000825	float
total_cloud_cover:p	72	0.000825	float
visibility:m	72	0.000825	float
wind_speed_10m:ms	72	0.000825	float
wind_speed_u_10m:ms	72	0.000825	float
wind_speed_v_10m:ms	72	0.000825	float
y			float

	variable_type	special_types
absolute_humidity_2m:gm3	numeric	
air_density_2m:kgm3	numeric	
ceiling_height_agl:m	numeric	
clear_sky_energy_1h:J	numeric	
clear_sky_rad:W	numeric	
cloud_base_agl:m	numeric	
dew_point_2m:K	numeric	
diffuse_rad:W	numeric	
diffuse_rad_1h:J	numeric	
direct_rad:W	numeric	
direct_rad_1h:J	numeric	
effective_cloud_cover:p	numeric	
elevation:m	category	
fresh_snow_1h:cm	numeric	
fresh_snow_3h:cm	numeric	
fresh_snow_6h:cm	numeric	
is_day:idx	category	
is_estimated	category	
is_in_shadow:idx	category	
location	category	

msl_pressure:hPa	numeric
precip_5min:mm	numeric
precip_type_5min:idx	category
pressure_100m:hPa	numeric
pressure_50m:hPa	numeric
rain_water:kgm2	numeric
relative_humidity_1000hPa:p	numeric
sample_weight	category
sfc_pressure:hPa	numeric
snow_depth:cm	numeric
snow_melt_10min:mm	numeric
snow_water:kgm2	numeric
sun_azimuth:d	numeric
sun_elevation:d	numeric
super_cooled_liquid_water:kgm2	numeric
t_1000hPa:K	numeric
total_cloud_cover:p	numeric
visibility:m	numeric
wind_speed_10m:ms	numeric
wind_speed_u_10m:ms	numeric
wind_speed_v_10m:ms	numeric
y	numeric

test_data dataset summary

	count	unique	top	freq	mean \
absolute_humidity_2m:gm3	5791	289			4.192639
air_density_2m:kgm3	5791	640			1.280018
ceiling_height_agl:m	4395	4247			3278.267059
clear_sky_energy_1h:J	5788	3059			469132.824948
clear_sky_rad:W	5791	2046			130.246477
cloud_base_agl:m	4934	4719			1733.271034
dew_point_2m:K	5791	948			270.733081
diffuse_rad:W	5791	2237			42.175259
diffuse_rad_1h:J	5788	3065			152461.828645
direct_rad:W	5791	1829			51.829421
direct_rad_1h:J	5788	2676			186526.762509
effective_cloud_cover:p	5791	2100			66.598541
elevation:m	5791	3			11.262131
fresh_snow_1h:cm	5788	23			0.032308
fresh_snow_3h:cm	5788	42			0.100259
fresh_snow_6h:cm	5788	60			0.204492
is_day:idx	5791	5			0.488387
is_estimated	5791	1			1.0
is_in_shadow:idx	5791	5			0.555085
location	5791	3	A	2161	
msl_pressure:hPa	5791	2040			1012.678587
precip_5min:mm	5791	63			0.003687
precip_type_5min:idx	5791	12			0.086039

pressure_100m:hPa	5791	2124		998.781639
pressure_50m:hPa	5791	2134		1005.02648
rain_water:kgm2	5791	7		0.000984
relative_humidity_1000hPa:p	5791	2051		70.810205
sample_weight	5791	1		1.0
sfc_pressure:hPa	5791	2148		1011.29959
snow_depth:cm	5791	78		0.131661
snow_melt_10min:mm	5791	38		0.000695
snow_water:kgm2	5791	68		0.078393
sun_azimuth:d	5791	5681		179.475343
sun_elevation:d	5791	5093		-0.927197
super_cooled_liquid_water:kgm2	5791	31		0.035175
t_1000hPa:K	5791	825		275.185991
total_cloud_cover:p	5791	1838		71.785616
visibility:m	5791	5784		29884.461577
wind_speed_10m:ms	5791	424		3.227599
wind_speed_u_10m:ms	5791	672		0.668019
wind_speed_v_10m:ms	5791	483		0.538344
y	5791	2304		272.991992

	std	min	25%	\
absolute_humidity_2m:gm3	1.300644	1.1	3.35	
air_density_2m:kgm3	0.024372	1.219	1.26375	
ceiling_height_agl:m	2590.751931	27.925	1149.0625	
clear_sky_energy_1h:J	689638.596662	0.0	0.0	
clear_sky_rad:W	191.578221	0.0	0.0	
cloud_base_agl:m	1987.046511	27.5	525.4375	
dew_point_2m:K	4.634046	255.05	268.33749	
diffuse_rad:W	59.158733	0.0	0.0	
diffuse_rad_1h:J	211011.771342	0.0	0.0	
direct_rad:W	110.450287	0.0	0.0	
direct_rad_1h:J	393513.65175	0.0	0.0	
effective_cloud_cover:p	37.583548	0.0	33.6375	
elevation:m	7.8114	6.0	6.0	
fresh_snow_1h:cm	0.170919	0.0	0.0	
fresh_snow_3h:cm	0.425766	0.0	0.0	
fresh_snow_6h:cm	0.738932	0.0	0.0	
is_day:idx	0.486436	0.0	0.0	
is_estimated	0.0	1.0	1.0	
is_in_shadow:idx	0.483636	0.0	0.0	
location				
msl_pressure:hPa	13.953847	975.3	1003.875	
precip_5min:mm	0.017701	0.0	0.0	
precip_type_5min:idx	0.393918	0.0	0.0	
pressure_100m:hPa	13.825369	962.4	989.9	
pressure_50m:hPa	13.873049	968.45	996.087475	
rain_water:kgm2	0.009596	0.0	0.0	
relative_humidity_1000hPa:p	14.940249	21.325	60.75	

```
sample_weight                          0.0        1.0         1.0
sfc_pressure:hPa                 13.921629     974.55     1002.25
snow_depth:cm                     0.635847        0.0         0.0
snow_melt_10min:mm                0.007333        0.0         0.0
snow_water:kgm2                   0.189057        0.0         0.0
sun_azimuth:d                    96.891969     14.913   94.264625
sun_elevation:d                  20.775858  -44.28175  -17.109625
super_cooled_liquid_water:kgm2    0.084895        0.0         0.0
t_1000hPa:K                       3.823552    261.975       272.8
total_cloud_cover:p              37.578218        0.0        41.8
visibility:m                  14669.627165     1215.4    18727.05
wind_speed_10m:ms                 1.869023       0.05       1.725
wind_speed_u_10m:ms                3.12501      -7.15       -1.75
wind_speed_v_10m:ms               1.838513       -5.3        -0.8
y                               770.841016       -0.0         0.0

                                     50%          75%         max   dtypes  \
absolute_humidity_2m:gm3             4.3         5.05         7.7  float64
air_density_2m:kgm3                1.279      1.29375     1.37175  float64
ceiling_height_agl:m             2618.95     4661.025   12294.901  float64
clear_sky_energy_1h:J            11008.5     791394.0   2554290.5  float64
clear_sky_rad:W                    2.675      221.925       710.5  float64
cloud_base_agl:m                 904.825  2014.962525     10674.3  float64
dew_point_2m:K                     271.6        273.9       280.4  float64
diffuse_rad:W                      1.775      78.4875      311.95  float64
diffuse_rad_1h:J                 18860.9   279202.425   1071799.5  float64
direct_rad:W                         0.0      34.0875      530.15  float64
direct_rad_1h:J                      0.0     129529.5   1895533.0  float64
effective_cloud_cover:p           85.375       99.975       100.0  float64
elevation:m                          7.0         24.0        24.0  float64
fresh_snow_1h:cm                     0.0          0.0         2.6  float64
fresh_snow_3h:cm                     0.0          0.0         5.2  float64
fresh_snow_6h:cm                     0.0          0.0         7.5  float64
is_day:idx                          0.25          1.0         1.0  float64
is_estimated                         1.0          1.0         1.0    int64
is_in_shadow:idx                     1.0          1.0         1.0  float64
location                                                            object
msl_pressure:hPa                1011.625    1023.8125   1041.3501  float64
precip_5min:mm                       0.0          0.0      0.2475  float64
precip_type_5min:idx                 0.0          0.0         3.0  float64
pressure_100m:hPa                  997.9     1009.875     1028.05  float64
pressure_50m:hPa                  1004.1    1016.1625     1034.45  float64
rain_water:kgm2                      0.0          0.0       0.175  float64
relative_humidity_1000hPa:p         73.1       82.075        98.0  float64
sample_weight                        1.0          1.0         1.0    int64
sfc_pressure:hPa                 1010.35    1022.5125   1040.8501  float64
snow_depth:cm                        0.0          0.0         4.9  float64
snow_melt_10min:mm                   0.0          0.0        0.14  float64
```

snow_water:kgm2	0.0	0.1	2.15	float64
sun_azimuth:d	179.52899	263.49875	347.37848	float64
sun_elevation:d	-0.79825	15.30325	41.13025	float64
super_cooled_liquid_water:kgm2	0.0	0.0	0.75	float64
t_1000hPa:K	275.175	277.525	285.1	float64
total_cloud_cover:p	96.65	100.0	100.0	float64
visibility:m	31311.025	40438.6635	66178.45	float64
wind_speed_10m:ms	2.9	4.45	10.2	float64
wind_speed_u_10m:ms	0.3	2.9	9.95	float64
wind_speed_v_10m:ms	0.625	1.825	7.15	float64
y	0.0	142.906699	5172.64	float64

	missing_count	missing_ratio	raw_type	\
absolute_humidity_2m:gm3			float	
air_density_2m:kgm3			float	
ceiling_height_agl:m	1396	0.241064	float	
clear_sky_energy_1h:J	3	0.000518	float	
clear_sky_rad:W			float	
cloud_base_agl:m	857	0.147988	float	
dew_point_2m:K			float	
diffuse_rad:W			float	
diffuse_rad_1h:J	3	0.000518	float	
direct_rad:W			float	
direct_rad_1h:J	3	0.000518	float	
effective_cloud_cover:p			float	
elevation:m			float	
fresh_snow_1h:cm	3	0.000518	float	
fresh_snow_3h:cm	3	0.000518	float	
fresh_snow_6h:cm	3	0.000518	float	
is_day:idx			float	
is_estimated			int	
is_in_shadow:idx			float	
location			object	
msl_pressure:hPa			float	
precip_5min:mm			float	
precip_type_5min:idx			float	
pressure_100m:hPa			float	
pressure_50m:hPa			float	
rain_water:kgm2			float	
relative_humidity_1000hPa:p			float	
sample_weight			int	
sfc_pressure:hPa			float	
snow_depth:cm			float	
snow_melt_10min:mm			float	
snow_water:kgm2			float	
sun_azimuth:d			float	
sun_elevation:d			float	
super_cooled_liquid_water:kgm2			float	

t_1000hPa:K	float
total_cloud_cover:p	float
visibility:m	float
wind_speed_10m:ms	float
wind_speed_u_10m:ms	float
wind_speed_v_10m:ms	float
y	float

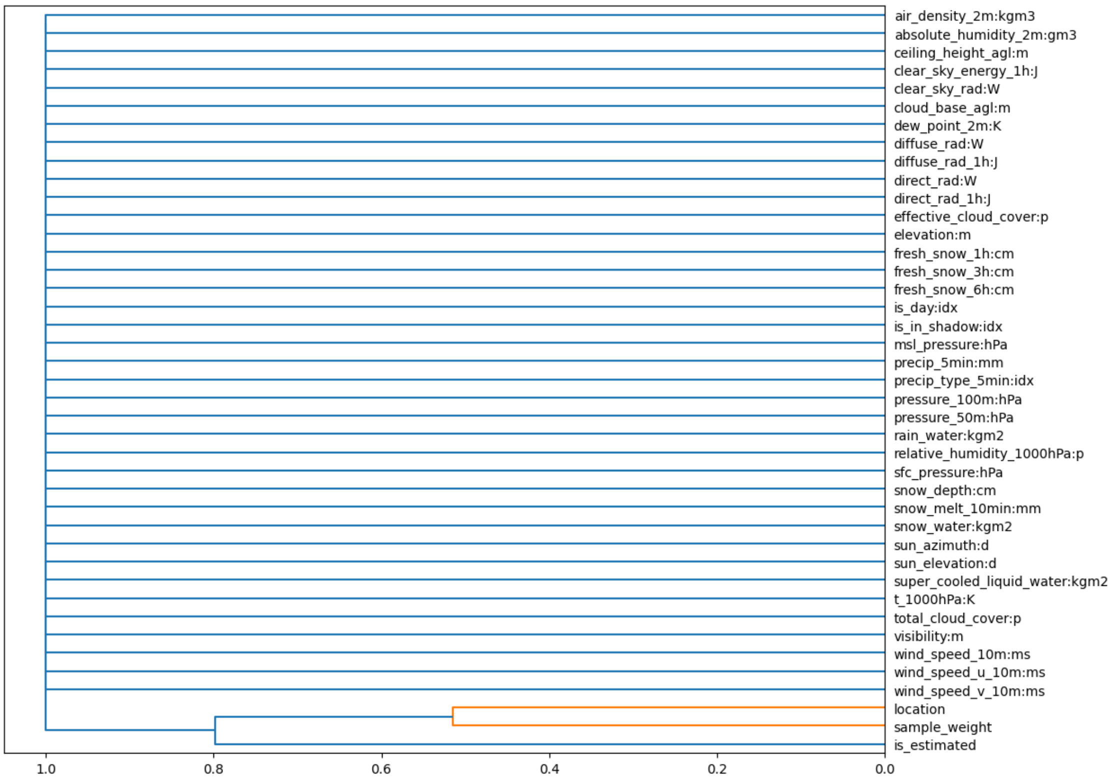
	variable_type	special_types
absolute_humidity_2m:gm3	numeric	
air_density_2m:kgm3	numeric	
ceiling_height_agl:m	numeric	
clear_sky_energy_1h:J	numeric	
clear_sky_rad:W	numeric	
cloud_base_agl:m	numeric	
dew_point_2m:K	numeric	
diffuse_rad:W	numeric	
diffuse_rad_1h:J	numeric	
direct_rad:W	numeric	
direct_rad_1h:J	numeric	
effective_cloud_cover:p	numeric	
elevation:m	category	
fresh_snow_1h:cm	numeric	
fresh_snow_3h:cm	numeric	
fresh_snow_6h:cm	numeric	
is_day:idx	category	
is_estimated	category	
is_in_shadow:idx	category	
location	category	
msl_pressure:hPa	numeric	
precip_5min:mm	numeric	
precip_type_5min:idx	category	
pressure_100m:hPa	numeric	
pressure_50m:hPa	numeric	
rain_water:kgm2	category	
relative_humidity_1000hPa:p	numeric	
sample_weight	category	
sfc_pressure:hPa	numeric	
snow_depth:cm	numeric	
snow_melt_10min:mm	numeric	
snow_water:kgm2	numeric	
sun_azimuth:d	numeric	
sun_elevation:d	numeric	
super_cooled_liquid_water:kgm2	numeric	
t_1000hPa:K	numeric	
total_cloud_cover:p	numeric	
visibility:m	numeric	
wind_speed_10m:ms	numeric	

```
wind_speed_u_10m:ms                     numeric
wind_speed_v_10m:ms                     numeric
y                                       numeric
```

Types warnings summary

```
              train_data test_data warnings
sample_weight      float       int  warning
```

1.0.1 Feature Distance

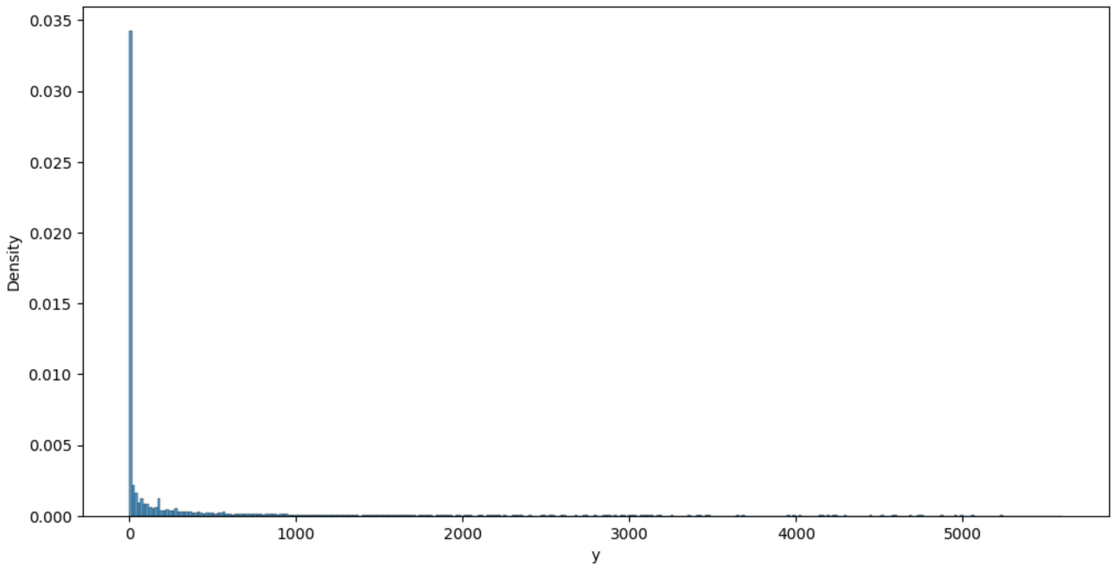


[6]:
```python
if run_analysis:
    auto.target_analysis(train_data=train_data, label="y")
```

1.1 Target variable analysis

```
   count        mean         std  min  25%  50%       75%      max   dtypes  \
y  10000  299.128516  787.495283  0.0  0.0  0.0  183.7125  5596.36  float64

   unique missing_count missing_ratio raw_type special_types
y    2419                                float
```

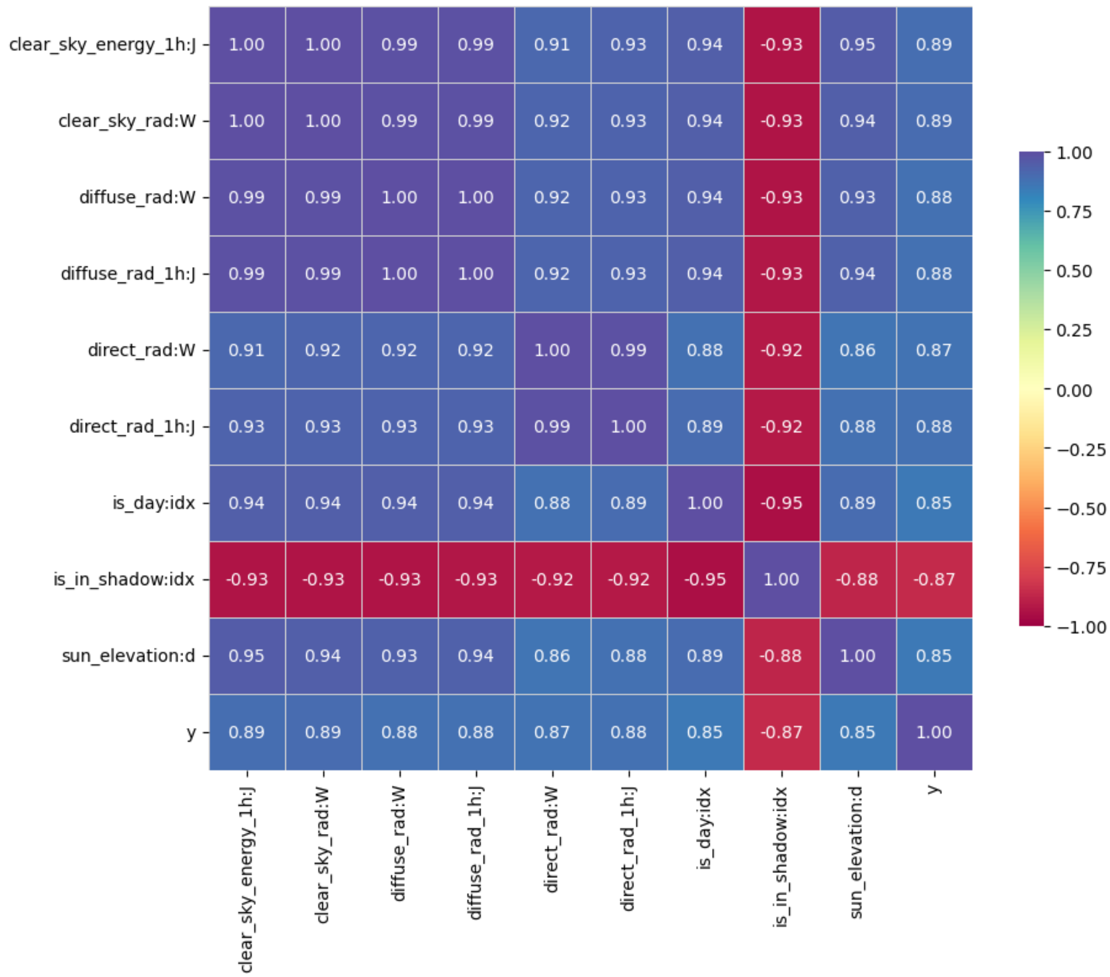


1.1.1 Distribution fits for target variable

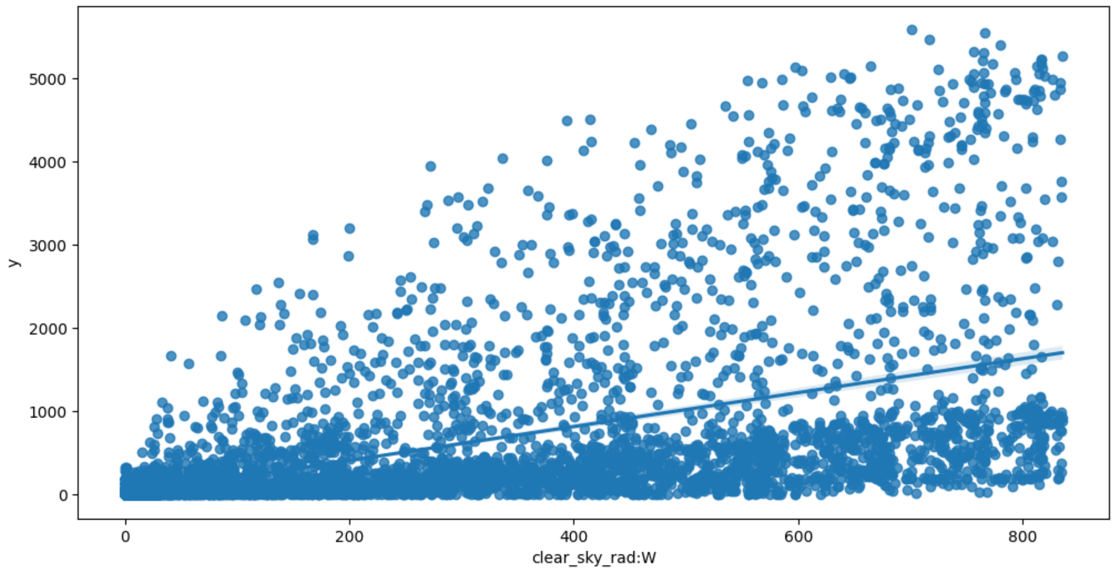
- none of the attempted distribution fits satisfy specified minimum p-value threshold: `0.01`

1.1.2 Target variable correlations

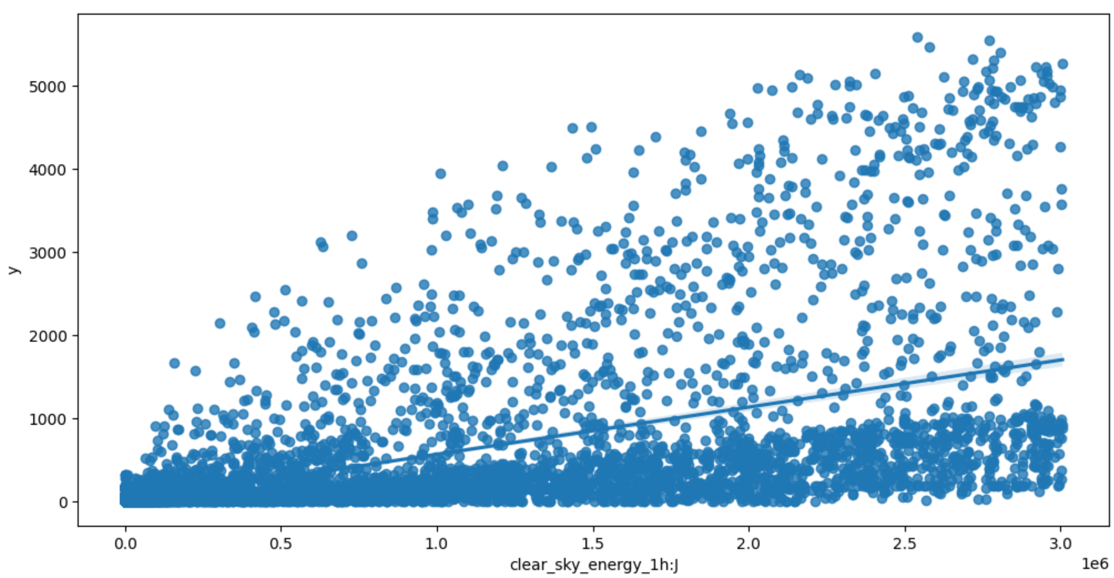
`train_data` - `spearman` correlation matrix; focus: absolute correlation for `y` >= `0.5` (sample size: 10000)



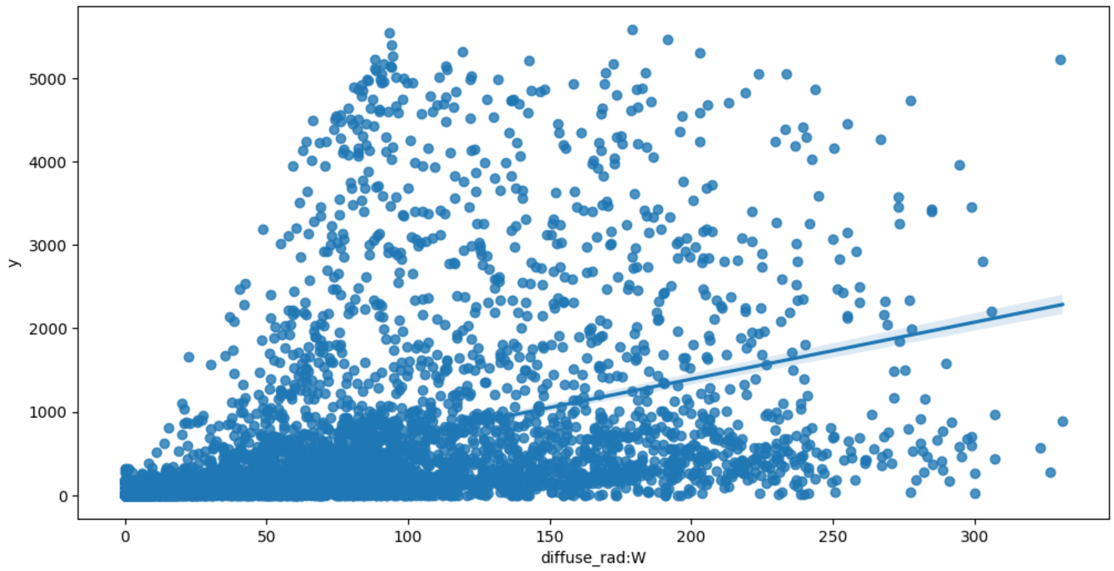
Feature interaction between `clear_sky_rad:W`/`y` in `train_data` (sample size: 10000)



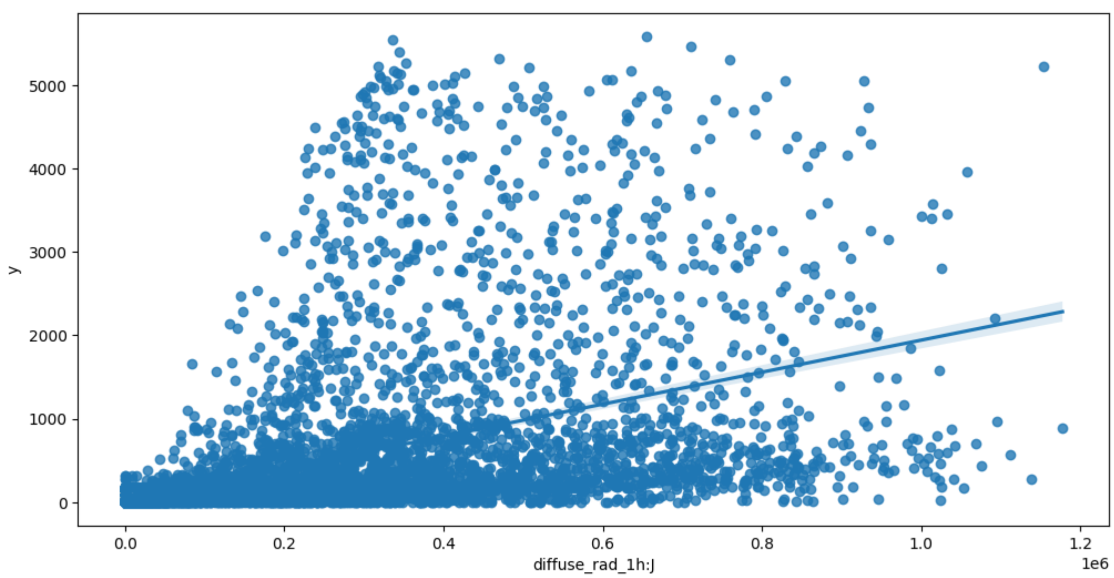
Feature interaction between `clear_sky_energy_1h:J/y` in `train_data` (sample size: 10000)



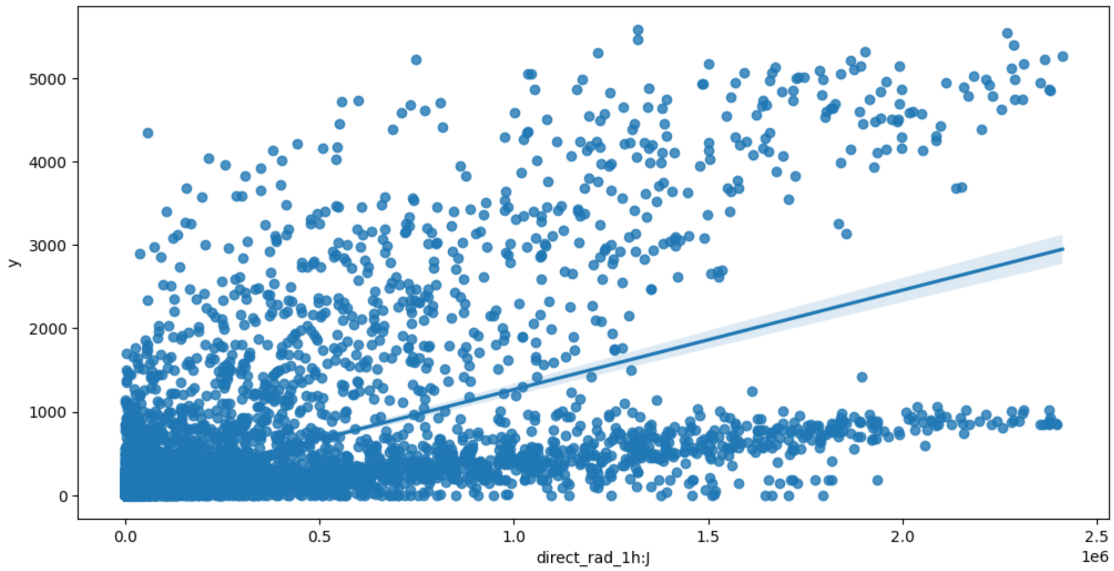
Feature interaction between `diffuse_rad:W/y` in `train_data` (sample size: 10000)



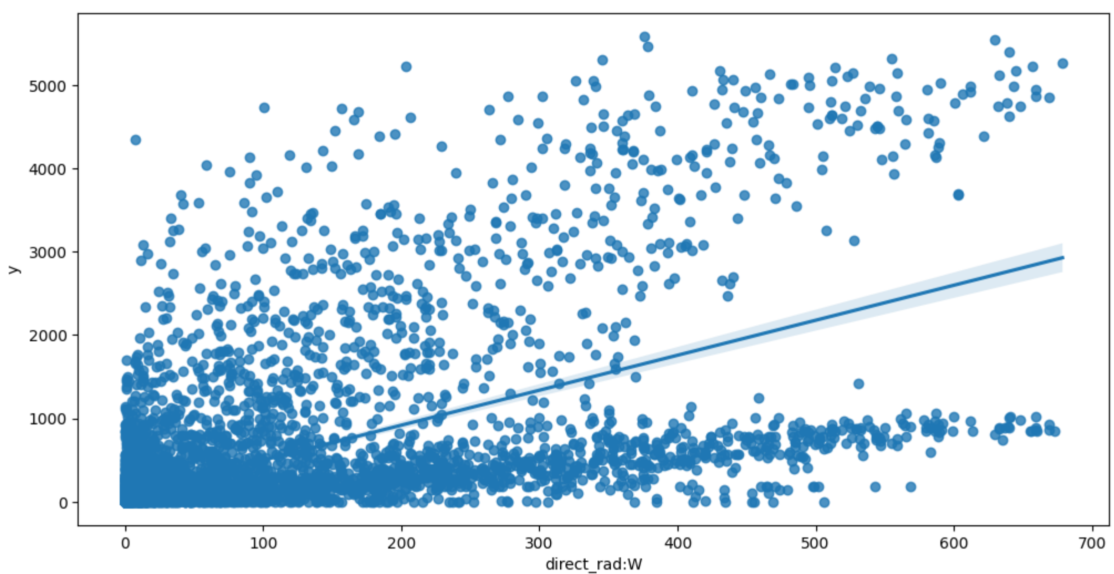
Feature interaction between `diffuse_rad_1h:J`/`y` in `train_data` (sample size: 10000)



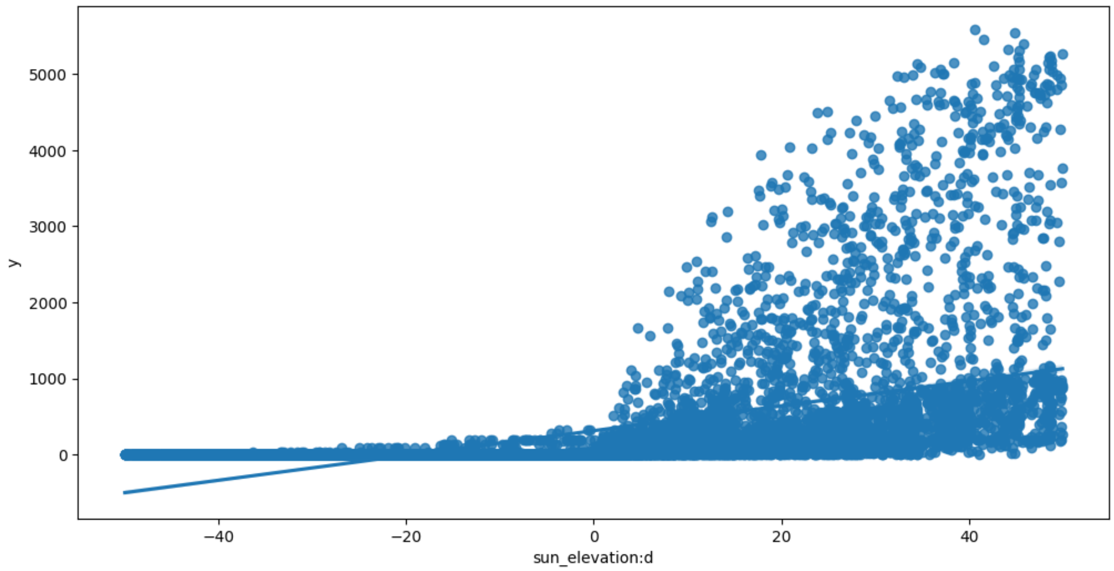
Feature interaction between `direct_rad_1h:J`/`y` in `train_data` (sample size: 10000)



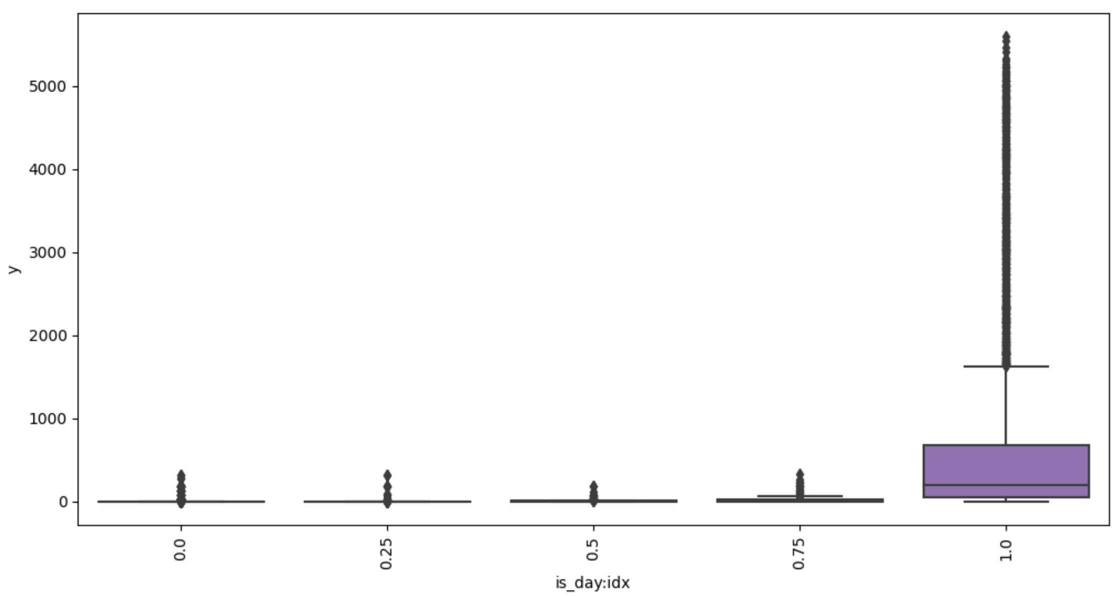
Feature interaction between `direct_rad:W`/y in `train_data` (sample size: 10000)



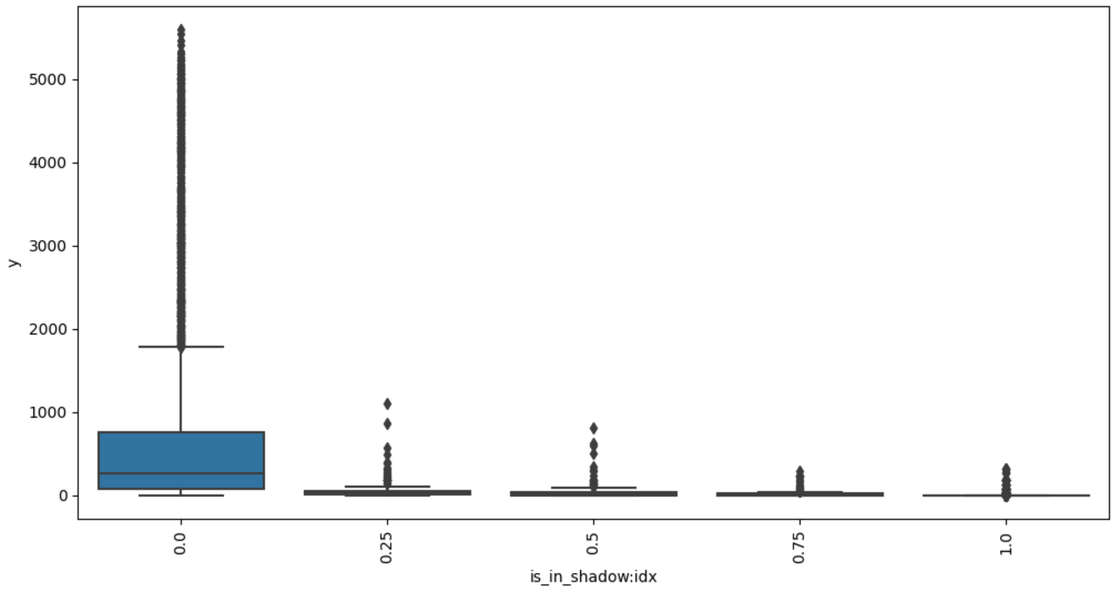
Feature interaction between `sun_elevation:d`/y in `train_data` (sample size: 10000)



Feature interaction between `is_day:idx/y` in `train_data` (sample size: 10000)



Feature interaction between `is_in_shadow:idx/y` in `train_data` (sample size: 10000)



2 Starting

[7]:
```
import os


# Get the last submission number
last_submission_number = int(max([int(filename.split('_')[1].split('.')[0]) for␣
  ↪filename in os.listdir('submissions') if "submission" in filename]))
print("Last submission number:", last_submission_number)
print("Now creating submission number:", last_submission_number + 1)

# Create the new filename
new_filename = f'submission_{last_submission_number + 1}'

hello = os.environ.get('HELLO')
if hello is not None:
    new_filename += f'_{hello}'

print("New filename:", new_filename)
```

```
Last submission number: 89
Now creating submission number: 90
New filename: submission_90
```

[8]:
```
predictors = [None, None, None]
```

```
[9]: def fit_predictor_for_location(loc):
    print(f"Training model for location {loc}...")
    # sum of sample weights for this location, and number of rows, for both␣
 ↪train and tune data and test data
    print("Train data sample weight sum:", train_data[train_data["location"] ==␣
 ↪loc]["sample_weight"].sum())
    print("Train data number of rows:", train_data[train_data["location"] ==␣
 ↪loc].shape[0])
    if use_tune_data:
        print("Tune data sample weight sum:",␣
 ↪tuning_data[tuning_data["location"] == loc]["sample_weight"].sum())
        print("Tune data number of rows:", tuning_data[tuning_data["location"]␣
 ↪== loc].shape[0])
    if use_test_data:
        print("Test data sample weight sum:", test_data[test_data["location"]␣
 ↪== loc]["sample_weight"].sum())
        print("Test data number of rows:", test_data[test_data["location"] ==␣
 ↪loc].shape[0])
    predictor = TabularPredictor(
        label=label,
        eval_metric=metric,
        path=f"AutogluonModels/{new_filename}_{loc}",
        sample_weight=sample_weight,
        weight_evaluation=weight_evaluation,
        groups="group" if use_groups else None,
    ).fit(
        train_data=train_data[train_data["location"] == loc],
        time_limit=time_limit,
        presets=presets,
        num_stack_levels=num_stack_levels,
        num_bag_folds=num_bag_folds if not use_groups else 2,# just put␣
 ↪somethin, will be overwritten anyways
        tuning_data=tuning_data[tuning_data["location"] == loc] if␣
 ↪use_tune_data else None,
        use_bag_holdout=use_bag_holdout,
        holdout_frac=holdout_frac,
    )

    # evaluate on test data
    if use_test_data:
        # drop sample_weight column
        t = test_data[test_data["location"] == loc]#.
 ↪drop(columns=["sample_weight"])
        perf = predictor.evaluate(t)
        print("Evaluation on test data:")
        print(perf[predictor.eval_metric.name])
```

```python
    return predictor

loc = "A"
predictors[0] = fit_predictor_for_location(loc)
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_90_A"
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Values in column 'sample_weight' used as sample weights instead of predictive
features. Evaluation will report weighted metrics, so ensure same column exists
in test data.
Beginning AutoGluon training … Time limit = 1800s
AutoGluon will save models to "AutogluonModels/submission_90_A/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Linux
Platform Machine:   x86_64
Platform Version:   #1 SMP Debian 5.10.197-1 (2023-09-29)
Disk Space Avail:   298.11 GB / 315.93 GB (94.4%)
Train Data Rows:    31924
Train Data Columns: 41
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (5733.42, 0.0, 632.68576,
1165.32372)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    131761.1 MB
        Train Data (Original)  Memory Usage: 11.81 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
```

```
                Fitting DropUniqueFeatureGenerator…
Training model for location A…
Train data sample weight sum: 31924.000000000007
Train data number of rows: 31924
Test data sample weight sum: 2161
Test data number of rows: 2161
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 1): ['location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 38 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 37 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['elevation:m', 'is_estimated']
        0.2s = Fit runtime
        39 features in original data used to generate 39 features in processed
data.
        Train Data (Processed) Memory Usage: 9.51 MB (0.0% of available memory)
Data preprocessing and feature engineering runtime = 0.18s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
```

```
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 1199.58s of
the 1799.81s of remaining time.
        -207.0745        = Validation score   (-mean_absolute_error)
        0.04s    = Training   runtime
        0.42s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 1198.94s of
the 1799.18s of remaining time.
        -208.1423        = Validation score   (-mean_absolute_error)
        0.04s    = Training   runtime
        0.4s     = Validation runtime
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 1198.44s of the
1798.68s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -139.7504        = Validation score   (-mean_absolute_error)
        38.16s   = Training   runtime
        21.85s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 1151.98s of the
1752.21s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -148.2262        = Validation score   (-mean_absolute_error)
        41.99s   = Training   runtime
        14.42s   = Validation runtime
Fitting model: RandomForestMSE_BAG_L1 … Training model for up to 1106.75s of
the 1706.98s of remaining time.
        -162.4541        = Validation score   (-mean_absolute_error)
        10.04s   = Training   runtime
        1.2s     = Validation runtime
Fitting model: CatBoost_BAG_L1 … Training model for up to 1093.46s of the
1693.7s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -155.3753        = Validation score   (-mean_absolute_error)
        204.38s  = Training   runtime
        0.13s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 … Training model for up to 888.02s of the
1488.25s of remaining time.
```

```
        -163.5846        = Validation score   (-mean_absolute_error)
        2.01s    = Training   runtime
        1.2s     = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 882.7s of the
1482.93s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -169.7354        = Validation score   (-mean_absolute_error)
        39.32s   = Training   runtime
        0.64s    = Validation runtime
Fitting model: XGBoost_BAG_L1 … Training model for up to 842.26s of the
1442.5s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -158.1199        = Validation score   (-mean_absolute_error)
        63.81s   = Training   runtime
        3.27s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 775.1s of the
1375.34s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -153.7348        = Validation score   (-mean_absolute_error)
        132.53s  = Training   runtime
        0.36s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 641.4s of the
1241.64s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -144.4866        = Validation score   (-mean_absolute_error)
        133.11s  = Training   runtime
        19.94s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 360.0s of the
1101.5s of remaining time.
        -136.91  = Validation score   (-mean_absolute_error)
        0.86s    = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 1100.62s of the
1100.6s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -139.2783        = Validation score   (-mean_absolute_error)
        3.0s     = Training   runtime
        0.17s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 1096.36s of the
1096.34s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
```

```
ParallelLocalFoldFittingStrategy
        -136.5524        = Validation score   (-mean_absolute_error)
        2.29s    = Training   runtime
        0.09s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 1092.84s of
the 1092.82s of remaining time.
        -135.9141        = Validation score   (-mean_absolute_error)
        15.92s   = Training   runtime
        1.27s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 1073.58s of the
1073.56s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.9547        = Validation score   (-mean_absolute_error)
        4.83s    = Training   runtime
        0.06s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L2 … Training model for up to 1067.58s of the
1067.56s of remaining time.
        -135.6456        = Validation score   (-mean_absolute_error)
        2.61s    = Training   runtime
        1.25s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 1061.6s of
the 1061.59s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.0584        = Validation score   (-mean_absolute_error)
        39.72s   = Training   runtime
        0.67s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 1020.58s of the
1020.57s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.1474        = Validation score   (-mean_absolute_error)
        3.02s    = Training   runtime
        0.12s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 1016.3s of the
1016.28s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -137.1077        = Validation score   (-mean_absolute_error)
        53.6s    = Training   runtime
        0.54s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 961.31s of the
961.29s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -137.4687        = Validation score   (-mean_absolute_error)
        7.9s     = Training   runtime
```

```
        0.25s    = Validation runtime
Repeating k-fold bagging: 2/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 952.03s of the
952.01s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -138.7235        = Validation score   (-mean_absolute_error)
        6.69s    = Training   runtime
        0.37s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 947.09s of the
947.07s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.2762        = Validation score   (-mean_absolute_error)
        4.75s    = Training   runtime
        0.19s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 943.34s of the
943.33s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.7681        = Validation score   (-mean_absolute_error)
        9.76s    = Training   runtime
        0.12s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 936.98s of
the 936.97s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.066         = Validation score   (-mean_absolute_error)
        79.58s   = Training   runtime
        1.33s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 895.67s of the
895.65s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.3834        = Validation score   (-mean_absolute_error)
        5.99s    = Training   runtime
        0.26s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 891.22s of the
891.21s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.3568        = Validation score   (-mean_absolute_error)
        112.01s  = Training   runtime
        1.05s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 831.46s of the
831.44s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
```

```
        -136.6957        = Validation score   (-mean_absolute_error)
        15.37s   = Training   runtime
        0.46s    = Validation runtime
Repeating k-fold bagging: 3/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 822.71s of the
822.69s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -138.6952        = Validation score   (-mean_absolute_error)
        10.4s    = Training   runtime
        0.54s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 817.7s of the
817.68s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.1345        = Validation score   (-mean_absolute_error)
        7.01s    = Training   runtime
        0.28s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 814.11s of the
814.09s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.6666        = Validation score   (-mean_absolute_error)
        14.61s   = Training   runtime
        0.18s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 807.97s of
the 807.96s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.7458        = Validation score   (-mean_absolute_error)
        119.65s  = Training   runtime
        1.97s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 766.57s of the
766.55s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.2141        = Validation score   (-mean_absolute_error)
        8.89s    = Training   runtime
        0.39s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 762.35s of the
762.33s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.9574        = Validation score   (-mean_absolute_error)
        175.66s  = Training   runtime
        1.69s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 697.32s of the
697.31s of remaining time.
```

```
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.3156        = Validation score   (-mean_absolute_error)
        23.51s   = Training   runtime
        0.66s    = Validation runtime
Repeating k-fold bagging: 4/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 687.88s of the
687.87s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -138.5887        = Validation score   (-mean_absolute_error)
        14.25s   = Training   runtime
        0.73s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 682.66s of the
682.65s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.0618        = Validation score   (-mean_absolute_error)
        9.28s    = Training   runtime
        0.37s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 679.21s of the
679.2s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.5893        = Validation score   (-mean_absolute_error)
        19.79s   = Training   runtime
        0.24s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 672.76s of
the 672.74s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.5501        = Validation score   (-mean_absolute_error)
        160.12s  = Training   runtime
        2.7s     = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 630.86s of the
630.85s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.9661        = Validation score   (-mean_absolute_error)
        12.22s   = Training   runtime
        0.54s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 626.23s of the
626.22s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.7212        = Validation score   (-mean_absolute_error)
        239.1s   = Training   runtime
        2.18s    = Validation runtime
```

```
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 561.4s of the
561.39s of remaining time.
        Fitting 8 child models (S4F1 - S4F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.1661        = Validation score   (-mean_absolute_error)
        30.73s   = Training   runtime
        0.85s    = Validation runtime
Repeating k-fold bagging: 5/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 552.9s of the
552.88s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -138.5468        = Validation score   (-mean_absolute_error)
        18.09s   = Training   runtime
        0.92s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 547.72s of the
547.7s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.0882        = Validation score   (-mean_absolute_error)
        11.64s   = Training   runtime
        0.47s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 544.12s of the
544.1s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.5497        = Validation score   (-mean_absolute_error)
        25.51s   = Training   runtime
        0.3s     = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 537.1s of the
537.08s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.5143        = Validation score   (-mean_absolute_error)
        200.32s  = Training   runtime
        3.36s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 495.48s of the
495.46s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.9755        = Validation score   (-mean_absolute_error)
        15.13s   = Training   runtime
        0.66s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 491.1s of the
491.09s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.6579        = Validation score   (-mean_absolute_error)
```

```
        300.76s  = Training   runtime
        2.73s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 428.2s of the
428.18s of remaining time.
        Fitting 8 child models (S5F1 - S5F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.106         = Validation score   (-mean_absolute_error)
        38.7s    = Training   runtime
        1.07s    = Validation runtime
Repeating k-fold bagging: 6/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 418.88s of the
418.87s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -138.5034        = Validation score   (-mean_absolute_error)
        21.42s   = Training   runtime
        1.08s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 414.16s of the
414.15s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.0657        = Validation score   (-mean_absolute_error)
        14.05s   = Training   runtime
        0.56s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 410.49s of the
410.47s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.5486        = Validation score   (-mean_absolute_error)
        30.66s   = Training   runtime
        0.36s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 403.95s of
the 403.93s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.4519        = Validation score   (-mean_absolute_error)
        240.63s  = Training   runtime
        4.11s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 362.21s of the
362.19s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.9381        = Validation score   (-mean_absolute_error)
        17.98s   = Training   runtime
        0.8s     = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 357.82s of the
357.8s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
```

```
ParallelLocalFoldFittingStrategy
        -135.6084        = Validation score   (-mean_absolute_error)
        366.09s  = Training   runtime
        3.24s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 291.08s of the
291.06s of remaining time.
        Fitting 8 child models (S6F1 - S6F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.0691        = Validation score   (-mean_absolute_error)
        47.02s   = Training   runtime
        1.3s     = Validation runtime
Repeating k-fold bagging: 7/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 281.45s of the
281.43s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -138.4553        = Validation score   (-mean_absolute_error)
        25.67s   = Training   runtime
        1.29s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 275.97s of the
275.96s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.0951        = Validation score   (-mean_absolute_error)
        16.48s   = Training   runtime
        0.65s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 272.33s of the
272.32s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.5204        = Validation score   (-mean_absolute_error)
        35.88s   = Training   runtime
        0.42s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 265.75s of
the 265.73s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -134.341         = Validation score   (-mean_absolute_error)
        281.2s   = Training   runtime
        4.77s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 223.79s of the
223.78s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.0242        = Validation score   (-mean_absolute_error)
        21.05s   = Training   runtime
        0.93s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 219.29s of the
```

```
219.27s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -135.5437        = Validation score   (-mean_absolute_error)
        429.87s  = Training   runtime
        3.8s     = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 154.06s of the
154.05s of remaining time.
        Fitting 8 child models (S7F1 - S7F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -136.067         = Validation score   (-mean_absolute_error)
        54.33s   = Training   runtime
        1.48s    = Validation runtime
Completed 7/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 360.0s of the
145.45s of remaining time.
        -132.8711        = Validation score   (-mean_absolute_error)
        0.73s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 1655.32s … Best model:
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_90_A/")
WARNING: eval_metric='pearsonr' does not support sample weights so they will be
ignored in reported metric.
Evaluation: mean_absolute_error on test data: -186.24721157826426
        Note: Scores are always higher_is_better. This metric score can be
multiplied by -1 to get the metric value.
Evaluations on test data:
{
    "mean_absolute_error": -186.24721157826426,
    "root_mean_squared_error": -420.62416270646276,
    "mean_squared_error": -176924.68625251285,
    "r2": 0.8716298559177108,
    "pearsonr": 0.9358825898767282,
    "median_absolute_error": -3.769514799118042
}

Evaluation on test data:
-186.24721157826426
```

[ ]:
```
loc = "B"
predictors[1] = fit_predictor_for_location(loc)
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_90_B"
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
```

```
Values in column 'sample_weight' used as sample weights instead of predictive
features. Evaluation will report weighted metrics, so ensure same column exists
in test data.
Beginning AutoGluon training … Time limit = 1800s
AutoGluon will save models to "AutogluonModels/submission_90_B/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Linux
Platform Machine:   x86_64
Platform Version:   #1 SMP Debian 5.10.197-1 (2023-09-29)
Disk Space Avail:   297.94 GB / 315.93 GB (94.3%)
Train Data Rows:    30792
Train Data Columns: 41
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 97.67477, 195.03642)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    130207.91 MB
        Train Data (Original)  Memory Usage: 11.39 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…

Training model for location B…
Train data sample weight sum: 30792.0
Train data number of rows: 30792
Test data sample weight sum: 2051
Test data number of rows: 2051

        Useless Original Features (Count: 1): ['location']
                These features carry no predictive signal and should be manually
investigated.
```

```
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 38 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 37 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['elevation:m', 'is_estimated']
        0.2s = Fit runtime
        39 features in original data used to generate 39 features in processed
data.
        Train Data (Processed) Memory Usage: 9.18 MB (0.0% of available memory)
Data preprocessing and feature engineering runtime = 0.18s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 1199.58s of
the 1799.81s of remaining time.
```

```
        -45.0801         = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.43s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 1199.06s of
the 1799.29s of remaining time.
        -44.9883         = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.43s    = Validation runtime
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 1198.55s of the
1798.78s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -26.235  = Validation score   (-mean_absolute_error)
        38.79s   = Training   runtime
        20.81s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 1155.96s of the
1756.2s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -27.6872         = Validation score   (-mean_absolute_error)
        44.86s   = Training   runtime
        16.33s   = Validation runtime
Fitting model: RandomForestMSE_BAG_L1 … Training model for up to 1107.7s of
the 1707.94s of remaining time.
        -31.8894         = Validation score   (-mean_absolute_error)
        11.38s   = Training   runtime
        1.17s    = Validation runtime
Fitting model: CatBoost_BAG_L1 … Training model for up to 1093.5s of the
1693.74s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -29.4372         = Validation score   (-mean_absolute_error)
        206.59s  = Training   runtime
        0.11s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 … Training model for up to 885.71s of the
1485.94s of remaining time.
        -32.6592         = Validation score   (-mean_absolute_error)
        1.93s    = Training   runtime
        1.16s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 880.85s of
the 1481.08s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -35.7575         = Validation score   (-mean_absolute_error)
        38.29s   = Training   runtime
        0.6s     = Validation runtime
Fitting model: XGBoost_BAG_L1 … Training model for up to 841.26s of the
1441.5s of remaining time.
```

```
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -30.4775         = Validation score   (-mean_absolute_error)
        105.3s   = Training   runtime
        24.76s   = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 731.93s of the
1332.17s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -28.9112         = Validation score   (-mean_absolute_error)
        198.41s  = Training   runtime
        0.46s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 532.27s of the
1132.5s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -26.5707         = Validation score   (-mean_absolute_error)
        139.44s  = Training   runtime
        26.82s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 360.0s of the
985.81s of remaining time.
        -25.5137         = Validation score   (-mean_absolute_error)
        0.82s    = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 984.97s of the
984.95s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -23.1597         = Validation score   (-mean_absolute_error)
        10.07s   = Training   runtime
        0.69s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 973.33s of the
973.31s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.8809         = Validation score   (-mean_absolute_error)
        2.81s    = Training   runtime
        0.12s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 969.32s of
the 969.3s of remaining time.
        -21.959  = Validation score   (-mean_absolute_error)
        15.03s   = Training   runtime
        1.19s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 952.57s of the
952.55s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
```

```
ParallelLocalFoldFittingStrategy
        -22.9341         = Validation score   (-mean_absolute_error)
        37.75s   = Training   runtime
        0.06s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L2 … Training model for up to 913.65s of the
913.64s of remaining time.
        -21.9244         = Validation score   (-mean_absolute_error)
        2.39s    = Training   runtime
        1.22s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 909.48s of
the 909.47s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.1783         = Validation score   (-mean_absolute_error)
        38.24s   = Training   runtime
        0.63s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 869.92s of the
869.91s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.6477         = Validation score   (-mean_absolute_error)
        3.04s    = Training   runtime
        0.13s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 865.47s of the
865.46s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.2668         = Validation score   (-mean_absolute_error)
        89.3s    = Training   runtime
        0.6s     = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 774.85s of the
774.84s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.4653         = Validation score   (-mean_absolute_error)
        98.41s   = Training   runtime
        1.73s    = Validation runtime
Repeating k-fold bagging: 2/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 671.26s of the
671.25s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -23.0488         = Validation score   (-mean_absolute_error)
        17.41s   = Training   runtime
        1.12s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 662.56s of the
662.55s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
```

```
ParallelLocalFoldFittingStrategy
        -22.8141         = Validation score   (-mean_absolute_error)
        5.29s    = Training   runtime
        0.23s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 658.7s of the
658.69s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.8401         = Validation score   (-mean_absolute_error)
        48.76s   = Training   runtime
        0.12s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 646.5s of the
646.48s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.0293         = Validation score   (-mean_absolute_error)
        76.47s   = Training   runtime
        1.31s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 606.94s of the
606.92s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.5148         = Validation score   (-mean_absolute_error)
        6.21s    = Training   runtime
        0.26s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 602.47s of the
602.45s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.0278         = Validation score   (-mean_absolute_error)
        189.07s  = Training   runtime
        1.11s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 501.31s of the
501.3s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.2095         = Validation score   (-mean_absolute_error)
        204.76s  = Training   runtime
        4.79s    = Validation runtime
Repeating k-fold bagging: 3/20
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 389.89s of the
389.88s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -23.0258         = Validation score   (-mean_absolute_error)
        24.06s   = Training   runtime
        1.58s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 381.85s of the
```

```
381.83s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.7665         = Validation score   (-mean_absolute_error)
        8.3s     = Training   runtime
        0.35s    = Validation runtime
Fitting model: CatBoost_BAG_L2 … Training model for up to 377.65s of the
377.63s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.7782         = Validation score   (-mean_absolute_error)
        62.72s   = Training   runtime
        0.19s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 … Training model for up to 362.45s of
the 362.43s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -21.9458         = Validation score   (-mean_absolute_error)
        115.45s  = Training   runtime
        1.97s    = Validation runtime
Fitting model: XGBoost_BAG_L2 … Training model for up to 322.19s of the
322.17s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.4749         = Validation score   (-mean_absolute_error)
        9.33s    = Training   runtime
        0.39s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L2 … Training model for up to 317.71s of the
317.69s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -21.9414         = Validation score   (-mean_absolute_error)
        270.03s  = Training   runtime
        1.62s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 … Training model for up to 235.42s of the
235.4s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -22.1519         = Validation score   (-mean_absolute_error)
        296.67s  = Training   runtime
        6.16s    = Validation runtime
Completed 3/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 360.0s of the
139.58s of remaining time.
        -21.4202         = Validation score   (-mean_absolute_error)
        0.7s     = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 1661.17s … Best model:
```

```
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_90_B/")
```

[ ]:
```
loc = "C"
predictors[2] = fit_predictor_for_location(loc)
```

3 Submit

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

train_data_with_dates = TabularDataset('X_train_raw.csv')
train_data_with_dates["ds"] = pd.to_datetime(train_data_with_dates["ds"])

test_data = TabularDataset('X_test_raw.csv')
test_data["ds"] = pd.to_datetime(test_data["ds"])
#test_data
```

[ ]:
```
test_ids = TabularDataset('test.csv')
test_ids["time"] = pd.to_datetime(test_ids["time"])
# merge test_data with test_ids
test_data_merged = pd.merge(test_data, test_ids, how="inner", right_on=["time",␣
↪"location"], left_on=["ds", "location"])

#test_data_merged
```

[ ]:
```
# predict, grouped by location
predictions = []
location_map = {
    "A": 0,
    "B": 1,
    "C": 2
}
for loc, group in test_data.groupby('location'):
    i = location_map[loc]
    subset = test_data_merged[test_data_merged["location"] == loc].
↪reset_index(drop=True)
    #print(subset)
    pred = predictors[i].predict(subset)
    subset["prediction"] = pred
    predictions.append(subset)

    # get past predictions
    past_pred = predictors[i].
↪predict(train_data_with_dates[train_data_with_dates["location"] == loc])
```

```
    train_data_with_dates.loc[train_data_with_dates["location"] == loc,␣
 ↪"prediction"] = past_pred
```

[ ]:
```
# plot predictions for location A, in addition to train data for A
for loc, idx in location_map.items():
    fig, ax = plt.subplots(figsize=(20, 10))
    # plot train data
    train_data_with_dates[train_data_with_dates["location"]==loc].plot(x='ds',␣
 ↪y='y', ax=ax, label="train data")

    # plot predictions
    predictions[idx].plot(x='ds', y='prediction', ax=ax, label="predictions")

    # plot past predictions
    train_data_with_dates[train_data_with_dates["location"]==loc].plot(x='ds',␣
 ↪y='prediction', ax=ax, label="past predictions")

    # title
    ax.set_title(f"Predictions for location {loc}")
```

[ ]:
```
# concatenate predictions
submissions_df = pd.concat(predictions)
submissions_df = submissions_df[["id", "prediction"]]
submissions_df
```

[ ]:
```
# Save the submission DataFrame to submissions folder, create new name based on␣
 ↪last submission, format is submission_<last_submission_number + 1>.csv

# Save the submission
print(f"Saving submission to submissions/{new_filename}.csv")
submissions_df.to_csv(os.path.join('submissions', f"{new_filename}.csv"),␣
 ↪index=False)
print("jall1a")
```

[ ]:
```
# save this running notebook
from IPython.display import display, Javascript
import time

# hei123

display(Javascript("IPython.notebook.save_checkpoint();"))

time.sleep(3)
```

[ ]:
```
# save this notebook to submissions folder
import subprocess
```

```
import os
subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
 ↪join('notebook_pdfs', f"{new_filename}.pdf"), "autogluon_each_location.
 ↪ipynb"])
```

[ ]:
```
# feature importance
location="A"
split_time = pd.Timestamp("2022-10-28 22:00:00")
estimated = train_data_with_dates[train_data_with_dates["ds"] >= split_time]
estimated = estimated[estimated["location"] == location]
predictors[0].feature_importance(feature_stage="original", data=estimated,␣
 ↪time_limit=60*10)
```

[ ]:
```
# feature importance
observed = train_data_with_dates[train_data_with_dates["ds"] < split_time]
observed = observed[observed["location"] == location]
predictors[0].feature_importance(feature_stage="original", data=observed,␣
 ↪time_limit=60*10)
```

[ ]:
```
display(Javascript("IPython.notebook.save_checkpoint();"))
time.sleep(3)

subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
 ↪join('notebook_pdfs', f"{new_filename}_with_feature_importance.pdf"),␣
 ↪"autogluon_each_location.ipynb"])
```

[ ]:
```
# import subprocess

# def execute_git_command(directory, command):
#     """Execute a Git command in the specified directory."""
#     try:
#         result = subprocess.check_output(['git', '-C', directory] + command,␣
 ↪stderr=subprocess.STDOUT)
#         return result.decode('utf-8').strip(), True
#     except subprocess.CalledProcessError as e:
#         print(f"Git command failed with message: {e.output.decode('utf-8').
 ↪strip()}")
#         return e.output.decode('utf-8').strip(), False

# git_repo_path = "."

# execute_git_command(git_repo_path, ['config', 'user.email',␣
 ↪'henrikskog01@gmail.com'])
# execute_git_command(git_repo_path, ['config', 'user.name', hello if hello is␣
 ↪not None else 'Henrik eller Jørgen'])

# branch_name = new_filename
```

```
# # add datetime to branch name
# branch_name += f"_{pd.Timestamp.now().strftime('%Y-%m-%d_%H-%M-%S')}"

# commit_msg = "run result"

# execute_git_command(git_repo_path, ['checkout', '-b',branch_name])

# # Navigate to your repo and commit changes
# execute_git_command(git_repo_path, ['add', '.'])
# execute_git_command(git_repo_path, ['commit', '-m',commit_msg])

# # Push to remote
# output, success = execute_git_command(git_repo_path, ['push',␣
↪'origin',branch_name])

# # If the push fails, try setting an upstream branch and push again
# if not success and 'upstream' in output:
#     print("Attempting to set upstream and push again...")
#     execute_git_command(git_repo_path, ['push', '--set-upstream',␣
↪'origin',branch_name])
#     execute_git_command(git_repo_path, ['push', 'origin', 'henrik_branch'])

# execute_git_command(git_repo_path, ['checkout', 'main'])
```